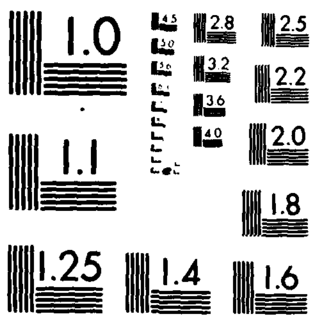
AD-A081 798 WHARTON SCHOOL PHILADELPHIA PA DEPT OF DECISION SCIENCES F/6 12/1
THE UNIQUENESS ASSUMPTION FOR FUNCTIONAL DEPENDENCIES.(U)
MAY 79 A BELLER N00014-75-C-0462
UNCLASSIFIED 79-06-03 NL

END
DATE
FILMED
4 80
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

Wharton

Department of Decision Sciences

THE UNIQUENESS ASSUMPTION FOR FUNCTIONAL DEPENDENCIES

AARON BELLER

79-06-03

(8)

University of
Pennsylvania
Philadelphia PA 19104

(8)

THE UNIQUENESS ASSUMPTION FOR FUNCTIONAL DEPENDENCIES

Aaron Beller

79-06-03

Department of Decision Sciences
The Wharton School
University of Pennsylvania
Philadelphia, PA 19104

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 79-06-03	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER 9
4. TITLE (and Subtitle) 6 THE UNIQUENESS ASSUMPTION FOR FUNCTIONAL DEPENDENCIES.		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER 14 79-06-03
7. AUTHOR(s) 10 Aaron/Beller		8. CONTRACT OR GRANT NUMBER(s) 15 N00014-75-C-0462
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Decision Sciences, The Wharton School, Philadelphia, PA 19104		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Task NR049-272
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research, Department of the Navy, 800 N. Quincy St., Arlington, VA 22217		12. REPORT DATE 11 May 1979 12 22
		13. NUMBER OF PAGES 18
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Approved for public release; distribution unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

relational data base schema, functional dependencies, 3rd normal form, uniqueness assumption

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The algorithm for synthesizing relational data base schema in the 3rd normal form assumes uniqueness of functional dependencies. This assumption is examined and a method for checking the assumption is presented.

402757

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT

The algorithm for synthesizing relational data base schema in the 3rd normal form assumes uniqueness of functional dependencies. This assumption is examined and a method for checking the assumption is presented.

I. INTRODUCTION

Beeri and Bernstein [2] have developed a fast algorithm for synthesizing relational data base schema in the 3rd normal form from a given set of FDs such that the resulting schema embodies the original FD's. They take an axiomatic approach to FDs and use a set of axiom schema to derive all the FDs that follow from a given set of FDs. If G is a given set of FDs, G+ denotes the set of derived FDs. In order for their algorithm to work they must make the following "uniqueness" assumption.

Let X be a set of attributes and A an attribute. If X-->A G+ then any derivation of X-->A represents the same "user intent".

In [2a] Beeri and Bernstein achieve the uniqunes assumption by assumming that all the relations of a data base are projections of a "universal relation". Since a universal relation never "really" exists the verification of the uniqunes assumption remains a problem.

When Beeri and Bernstein's algorithm is used to synthesize relational data base schema an attempt should be make to verify the uniqueness assumption. Even though Beeri and Bernstein's algorithm is linear any attempt to verify the uniqueness assumption is doomed to exponential time.

An automatic checking for violations of the uniqueness assumption is preferable to interactively "showing" each derivation to the user. Such a semantic analyzer is difficult to find since it is not known how to formalize the "user intent" of an FD. As a partial solution we classify FDs into three types, regular, injective and computable. Armstrong's [1] axioms can be applied to these types so that every derivation of an FD will result in classifying the FD as one of the three types (given the types of the initial FDs). When two derivations of an FD result in two different classifications then we have a violation. If two derivations both result in a calculable FD it is sometimes possible to decide that the calculations are different and there is a violation. In other cases it would not be known if there was a violation.

The usual solution to a violation of uniqueness is to rename some attributes. We show that this may lead to "problems" and that sometimes certain derivations must be "outlawed."

II. Preliminaries

We assume the reader is familiar with the notation and results in Beeri and Bernstein [2]. Our view of relational data bases is somewhat similar to that of Cadiou [4] and Nicholas [8]. There are two notions of a relation; intension and extension.

The intention of a relation should include as much of the "user intent" as possible. The intention of a relation consists of:

1. $R(A_1,\ldots,A_n)$ - a relational form, where R is the name of the relation and $\{A_1,\ldots,A_n\}$ are the attributes

2. A set of keys.

   Remark: 1. and 2. are usually called a relation scheme and in Beeri and Bernstein [2] this is all that is meant by the "intention."

3. Functional dependencies

4. Other types of dependencies

5. Domain definitions of the attributes

6. Other integrity constraints (See Eswarian [5] and Hammer Mcleod [7] for a taxonomy).

For each intension R there are many extensions. Each extension (or instance) of R is a finite set, R, of n-tuples satisfying the constraints of the intention.

The intension of a data base is a finite collection of relational intensions with additional integrity constraints (that include more than one relation). Attributes and their domain definitions are invarient over the data base so the FDs (and other dependencies) can be considered to reside in the data base as a whole.

The constraints on a data base can be stated in any appropriate language such as: first order predicate logic, SEQUEL, QUERY BY EXAMPLE. It is possible to discuss the set of all extentions of a data base but many questions (consistency, derivability) may be undecidable. For details consult Gallaire and Minker [6] and Nicholas [8].

The constraints for which the above questions are important are those that affect the structure of the data base. FDs affect the relational data base shcema since the normal forms are stated in terms of the FDs. Fortunately, under the uniqueness assumption, questions of consistency and derivability about FDs are decidable.

We shall review some notation from Beeri and Bernstein [2] and describe their program. An FD is denoted by X-->Y, where X and Y are sets of attributes. The only information that the above notation imparts is that for any relation $\underset{\sim}{R}$ whose attributes include $X \cup Y$ and for any instance R of $\underset{\sim}{R}$, if two tuples coincide on X they must also coincide on Y. Sometimes f:X-->Y is written, where f denotes a canonical name for the partial function from dom(X) to dom(Y) which is dependent on the extension (and changes as the extension does).

Armstrong [1] gave a set of axiom schema for deriving FDs from a given set of FDs and shows that the system is sound and complete. Beeri and Bernstein use the following equivalent axioms.

A_1:. (Reflexivity) X-->X

A_2:. (Augmentation) If X-->Z then $X \cup Y$-->Z

A_3:. (Pseudotransivity) If X-->Y and $Y \cup Z$-->W then $X \cup Z$-->W.

If G is a set of FDs, then G+ is the closure of G under the above axioms. An important part of Beeri and Bernstein's algorithm is to compute G+. If X-->Y can be derived from G it can be derived by an infinite number of derivations. By the uniqueness assumption Beeri and Bernstein can assume any derivation of X-->Y represents a unique "user intent." Hence they need only search for one such derivation. They only have to search derivation trees of height at most the number of attributes among the G since a derivation with a loop (i.e. one that goes through an attribute twice) is the same without the loop since X-->X must be the identity mapping by uniqueness.

III. Checking and Correcting the Uniqueness Assumption

Any method for verifying the uniqueness assumption will involve comparing different derivations of a single FD. By the above method, if X-->Y is not unique there are two derivations of X-->Y by trees of at most height twice the number of attributes (since at most one loop is needed). Hence if we had an algoritnm for deciding whether two derivations represent the same "user intent" the uniqueness problem would be decidable. Since we would have to search all derivations the solution is at best exponential. When a violation of uniqueness is discovered the usual remedy is to change attribute names so that two different FDs are produced. This would change the final relational data base shcema synthesized by the Beeri and Bernstein algorithm.

Primitively the "user" could be used as an oracle to decide whether derivations are unique. If a violation is found attributes can be named but it would produce more derivations and possibly violations. It is not clear that such a process terminates by some given bound (an example of this will be discussed later). Beeri and Bernstein suggest an alternative solution--simply reject some inferences. We shall see that this may be necessary.

IV. A Partial Solution

We propose a classification of FDs, with which we can automatically detect many of the violations of uniqueness. We define three types of FDs: regular, injective and computable.

Armstrong's axioms are adapted to include the above types. A violation can be detected if two derivations of X-->Y are found with different classifications. When two derivations of X-->Y are of the same type then a violation can only be discovered by a finer classification. When both derivations are computable then a finer classification can be made by checking if they are the same computation. Unfortunately the general problem of equivalence is undecidable (depending on the language used). The situation is similar to program verification and most computations encountered will be simple enough to compare. The use of computable attributes in a data base is questionable and will be discussed later.

1. Regular FDs: These are those defined by Armstrong.
   Example: EMP#--> Dept#; Dept#-->MGR#.

2. Injective (or One to One) FDs: This just means that the canonical function is always injective for each instance. We denote this by X<-->Y.
   Examples: X<-->X, SS#<-->Passport#.

3. Computable FDs: In this case the canonical function represents a real function of the attributes domains, other functions and the data base instance. We denote

this by F:X-->Y, where F represents the real function.

Notation: Lower case letters will denote the canonical names for non-computable FDs. Higher case letters will be reserved for computable FDs and general FDs (computable or non-computable) are denoted by Greek letters.

Examples:

A. F:SALARY, Number-of-Dependents-->WiDholding-Tax

B. If A and B are attributes we may have A+B=K a constant and G:A-->B where G=K-A

C. H: Dept# --->Number-of-Employees

The algorithm for H is to count the number of employees in the department for each instance.

One may ask if computable attributes should be in a data base altogether. The answer is that in general they should not. If the computation is cheap it can be recalculated every time there is a query. Even if not the attribute should be virtual in the following sense:

1. The attribute should be attached to an appropriate relation- but not be considered in the relational schema and should not take part in questions of the various forms (i.e., not used for FDs).

2. Every time an update is made that affects the value of the virtual attribute in a tuple, it should be recalculated.

3. It should be included among the attributes for queries.

The only problem this would present is for one to one computations F:A-->B and F^{-1}:B-->A (as is the case for A+B=K). Then we would have to decide which was more "basic" and this may not be known by the user. Hence in such cases it is better to leave them in and consider them for the normal forms. It seems that in practice computable FDs are included among the attributes of data bases even if they are not one-to-one. This is the case for some of the examples in Beeri and Bernstein [2].

It is easy to see that Armstrong's axioms can be adapted as follows:

A_1 a. X<-->X

b. if X<-->Y then Y<-->X

A_2 a. if X Z (or X<-->Z) then X ∪ Y-->Z

b. if F:X-->Z then F^*:X ∪ Y-->Z where F^*(X,Y)=F(X)

A_3 a. if [(X-->Y and Y ∪ Z-->W) or

(X<-->Y and Y ∪ Z-->W) or

(X-->Y and Y ∪ Z<-->W)] then

X ∪ Z-->W

b. if X<-->Y and Y ∪ Z<-->W then X ∪ Z<-->W

c. if $F:X\text{-->}Y$ and ($Y \cup Z\text{-->}W$ or $Y \cup Z\text{<-->}W$) then $F^*:X \cup Z\text{-->}W$ where if f is the canonical name for $Y \cup Z\text{-->}W$ ($Y \cup Z\text{<-->}W$) then $F^*(X,Z)=f(F(X),Z)$.

d. if ($X\text{-->}Y$ or $X\text{<-->}Y$) and $F:Y \cup Z\text{-->}W$ then $F^*:X \cup Z\text{-->}W$ where if f is the canonical name for $X\text{-->}Y$ ($X\text{<-->}Y$) then $F^*(X,Z)=F(f(X),Z)$

e. if $F_1:X\text{-->}Y$ and $F_2:Y \cup Z\text{-->}W$ then $F^*:X \cup Z\text{-->}W$ where $F^*(X,Z)=F_2(F_1(X), Z)$.

It is obvious that the correct classification is given in each case.

We start with a set of user classified FDs. It can be assumed that there are no violations among the given FDs. Careful specification of the FDs will help prevent violations. We shall illustrate how the above classification of FDs can be used to detect violations by using the three examples given in Beeri and Bernstein [2].

In order to define computable functions we need a function manipulation language (we cannot use relations since we are only given FDs). Buneman and Frankel [3] have developed a function query language which would more than suffice for our purposes. Since its syntax is not commonly known we will develop only sufficient tools for our examples intuitively.

Non-computable FDs (denoted by lower case letters) are treated as atoms. We allow composition of functions (this was already used in the adapted axioms). Let α :A-->B (remember Greek letters represent both computable and non-computable FDs). α is realized in an extension of the data base as a partial function α : dom(A)-->dom(B) since dom(A) may be infinite but only a finite number of values are realized in any extension. Let $\overline{\text{dom}(A)}$ represent the finite subset realized in a data base extension. Let α(A) represent the set $\{\alpha(a) \mid a \in \overline{\text{dom}(A)}\}$. For $b \in \overline{\text{dom}(B)}$, $\chi_{\#(A)=b}$ is the characteristic function of the predicate α(A)=b defined over dom(A), i.e., $\chi_{\#(A)=b}$: dom(A)-->{0,1} defined by:

$$\chi_{\#(A)=b}(a)=\begin{cases} 1 \text{ if } & (a)=b \\ 0 \text{ otherwise} & \end{cases}$$

We allow taking the sum of a set hence if f:Emp# -->Dept# then we can define a computable function F, F:Dept# -->Number-of-Employees by:

$$F(d)=\sum_{\text{dom}(Emp^{\#})} (\chi_{f(Emp^{\#})=d}) \text{ for } d \in \overline{\text{dom}(Dept^{\#})}.$$

Thus F would count the number of employees in a department.

Let g:Dept# -->Mgr#, we can define a computable G:Mgr# -->Number-of-Employees by:

$$G(m)=\sum_{\text{dom}(Dept^{\#})} [(\chi_{g(Dept^{\#})=m})*(F(Dept^{\#}))],$$

which computes the number of employees for a particular manager.

The above is just the embryo of a language but it is enough for our own purposes.

Example 1. We are given f_1:Dept# -->Mgr#, f_2:Emp# -->Dept#, Floor, F_3:Dept#, Floor-->Number-of-Employees, and F_4:Mgr#, Floor-->Number-of-Employees.

F_3 is computable by:

$$F_3(d,f) = \sum_{dom(Emp^{\#})} (\chi_{f_2(Emp^{\#})=d,f}).$$

F_4 is computable by:

$$F_4(m,f) = \sum_{dom(Dept^{\#})} [(\chi_{f_1(Dept^{\#})=m})*F_3(Dept^{\#},f)].$$

Using $A_3(d)$ on f_1:Dept# -->Mgr# and F_4:Mgr#, Floor-->Number-of-Employees we derive G:Dept#,Floor-->Number-of-Employees where,

$$G(d,f) = F_4(f_1(d),f) = \sum_{dom(Dept^{\#})} [(\chi_{f_1(Dept^{\#})=f_1(d)})*F(Dept^{\#},f)].$$

Clearly an algorithm could be defined which could decide $G \models F_3$; hence there are two derivations of Dept$^{\#}$, Floor-->Number-of-Employees with different user intents. Beeri and Bernstein's solution to this violation is to change F_4 to:

F_4:Mgr$^{\#}$,Floor-->Number-of-Employees-of-Manager

<u>Remark</u>. If we did not have F_2 (as is the case in the Beeri and Bernstein [2] example, F_3 would revert to a non-computable f_3 but F_4 would remain computable. G would be computable also and a violation would be detected because there were two derivations of Dept$^{\#}$, Floor-->Number-of-Employees one regular and the other computable.

<u>Example 2</u>. Let f_5:Emp$^{\#}$ -->Mgr$^{\#}$ and f_6:Mgr$^{\#}$ <-->Emp$^{\#}$. By A_1(b) applied to f_6 we derive g_1:Emp$^{\#}$ <-->Mgr$^{\#}$ and this gives two derivations of Emp$^{\#}$ -->Mgr$^{\#}$, one regular and the other injective. Here there is a violation. (We could have used transitivity on f_5, f_6 to get g_2:Emp$^{\#}$ -->Emp$^{\#}$ as opposed to the derivation of Emp$^{\#}$ <-->Emp$^{\#}$ by A_1(a).)

Beeri and Bernstein's solution to this violation is to change f_6 to f_6:Mgr$^{\#}$ -->Emp$^{\#}$-of-Mgr. Unfortunately this leads to additional problems. Assume we have a hierarchy of managers (manager of managers, etc.); how would the manager of a manager be determined. If the manager is treated as a regular employee in Emp$^{\#}$ -->Mgr$^{\#}$ an FD Emp$^{\#}$-of-Mgr<-->Emp$^{\#}$ is needed which again causes a violation. Otherwise a Emp$^{\#}$-of-Mgr->Mgr$^{\#}$-of-Mgr is needed, and so on until the highest manager. This is analogous to a geneology data base with a Son, Father relation. In such a case attributes for Grandfather, Greatgrandfather, etc. until Adam. This entails an explosion of attributes. The only feasible solution is to outlaw problematic derivations and consider Mgr -of-Mgr etc. a virtual computable.

<u>Example 3</u>. Let f_7:Stock# -->Store# and f_8:Stock#, Store# -->Qty. The "user intent" of f_7 is to map the Stock# onto Store# of the store that is in charge of ordering that item and f_8 maps Stock# and Store# of the store in which it is being sold into the quantity on hand. Using A_3(a) we derive g_3:Stock# -->Qty . Then A_2(a) gives g_4:Stock#, Store# -->Qty. f_8 and g_4 represent two different intents of Stock#, Store# -->Qty both classified regular. Hence they could not be distinguished by the classification method.

The above violation could be avoided by careful user definitions of FDs. The user should consider the range of the FD and if it does not include all the domain, a new attribute should be named. So Stock# -->Store doesn't include all store number in its range, only ordering store numbers.

IV. Discussion

It has been shown that checking the uniqueness assumption must be very time consuming. It is possible that certain limited types of sets of FDs cannot lead to violations but it is unlikely that these types could cover real situations.

Another problem encountered was a proliferation of attributes. Trying to satisfy the uniqueness assumption leads to many attribute names and it may make queries diffecult for a user if he has to differentiate Ordering Room , Storing Room , Personnel Room , etc.

In addition the uniqueness assumption cannot contend with "natural loops" as in the Emp# -->Mgr# -->Emp# etc.

It is interesting to investigate what happens to these semantic violations in a regular relational data base using a normal query language.

Since the formulation of a query uses relation names the "derivation" depends on how the query is presented. We shall use example 2 and SEQUEL to illustrate. Assume we have two relations E(<u>Emp</u>#, Mgr#) and M(<u>Mgr</u>#, Emp#).
If we wanted the Emp# of the manager of a given Emp#, e, we could write:

```
Select    Emp#
From      M
Where     Mgr# =
          Select    Mgr#
          From      E
          Where     Emp#=e
```

We could also form the equation on Mgr getting EXM which are the triples, (Emp#, Mgr#, Emp#), where the second Emp# is the Emp# of the manager. The natural join of E and M would be null because of the violation. In a regular relational data base such violations may cause some problems but are not catastrophic. If the above violations were not detected, the Beeri and Bernstein algorithm could eliminate E or M as redundant. Hence the desirability of synthesizing relational data base schema should be considered.

Bibliography

[1] Armstrong, "Dependency Structures of Data Base Relationships", Proc. IFIP 74, North Holland, 1974, pp. 580-583.

[2] Beeri and Bernstein, "An Algorithmic Approach to Normalization of Relation Database Schemas", Technical Report CSRG-73, Computer Science Research Group, University of Toronto, Sept., 1976.

[2a] Beeri and Bernstein,"Problems in Design of Normal Form Relational Schemas", ACM Transactions on Database Systems, March 1979, Volume 4 number 1, pg. 30-59.

[3] Buneman and Frankel, "FQL- A Functional Query Language" To Appear in SIGMOD 1979- Boston.

[4] Cadiou, "On Semantic Issues in the Relational Model of Data", Mathematical Foundations of Computer Science (A. Mazurkiewiz, Ed.), Vol. 45, Springer- Verlag, 1976, pp. 23-38.

[5] Eswarain and Chamberlain, "Functional Specifications of a Subsystem for Data Base Integrity", Proc. of the VLBD, Farmington, Mass. 1975, pp. 48-66.

[6] Gallaire and Minker, "Logic and Data Bases",Plenum Press, 1978.

[7] Hammer and Mcleod, "Semantic Integrity in a Relational Data Base System", Proc. of the VLBD, Farmington Mass. 1975, pp. 25-47.

[8] Nicolas, "First Order Logic Formilazation for Functional, Multivalued and Mutual Dependencies", Sigmod 1978, Austin, Texas, pp. 40-46.

DISTRIBUTION LIST

Department of the Navy - Office of Naval Research

Data Base Management Systems Project

Defense Documentation Center
(12 copies)
Cameron Station
Alexandria VA 22314

Office of Naval Research
(2 copies)
Information Systems Program
Code 437
Arlington VA 22217

Office of Naval Research
Code 102IP
Arlington Virginia 22217

Office of Naval Research
Branch Office
495 Summer Street
Boston MA 02210

Office of Naval Research
Branch Office Chicago
536 South Clark Street
Chicago IL 60605

Office of Naval Research
Branch Office Pasadena
1030 East Green Street
Pasadena CA 91106

New York Area Office

715 Broadway - 5th Floor
New York NY 10003

Naval Research Laboratory
(6 copies)
Technical Information Division
Code 2627
Washington DC 20375

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
(Code RD-1)
Washington DC 20380

Office of Naval Research
Code 455
Arlington VA 22217

Office of Naval Research
Code 458
Arlington VA 22217

Naval Electronics Laboratory Center
Advanced Software Technology Division
Code 5200
San Diego CA 92152

Mr. E. H. Gleissner
Naval Ship Research and
Development Center
Computation & Mathematics Dept.
Bethesda MD 20084

Mr. Kim B. Thompson
Technical Director
Information Systems Division
(OP-911G)
Office of Chief of Naval Operations
Washington DC 20350

Professor Omar Wing
Columbia University
in the City of New York
Dept. of Electrical Engineering
and Computer Science
New York NY 10027

Commander, Naval Sea Systems Command
Department of the Navy
Washington, D.C. 20362
ATTENTION (PMS30611)

Captain Richard L. Martin, USN
Commanding Officer
USS Francis Marion (LPA-249)
FPO New York 09501

Captain Grace M. Hopper
NAICOM/MIS Planning Branch
(OP-916D)
Office of Chief of Naval Operations
Washington DC 20350

Bureau of Library and
Information Science Research
Rutgers - The State University
189 College Avenue
New Brunswick NJ 08903
Attn: Dr. Henry Voos

Defense Mapping Agency
Topographic Center
ATTN: Advanced Technology
Division
Code 41300 (Mr. W. Mullison)
6500 Brookes Lane
Washington D.C. 20315

Major J.P. Pennell
Headquarters Marine Corps
Washington, D.C. 20380
ATTENTION: Code CCA-40

Professor Mike Athans
Massachusetts Institute of Technology
Dept. of Electrical Engineering and
Computer Science
77 Mass. Avenue
Cambridge, MA 02139